High Performance Computing Software: Achievements and Challenges

Ken Kennedy
Center for Research on Parallel Computation
Rice University

1. High Performance Computing and Communications Program
   - diversity of parallel architectures
2. Challenges for Software Support
   - performance, portability, usability
3. Future Architectures and Applications
   - petaflops architectures
   - distributed heterogeneous computer systems
   - new programming challenges

Status of Scalable Parallelism

- **Dream**
  - virtually limitless computing power at low cost
  - performance scalable from one to thousands of processors
    - applications would be programmed to scale automatically
  - easy portable programming
- **Reality**
  - successful at only moderate levels of scalability
    - state of the art: up to 32-way multiprocessor workstations
  - modest progress in programmability and scalability
  - usage primarily in research
  - limited penetration in industry
    - independent software vendors (ISVs) still reluctant
    - limited protection of programming investment

Trends in Architecture

- **Ascendancy of workstation and desktop technology**
  - shared-memory multiprocessors
  - multiprocessor personal computers
- **Deeper memory hierarchies**
  - Petaflops: may go to 10 levels
- **Distributed shared memory**
  - microprocessors moving to 64 bit addressing
  - desire to make maximum use of previous advances in cache structure
- **Clusters of workstations or PCs**
  - DSM and message-passing
- **Heterogeneous networks**
  - nodes of different power, architecture, data representation
  - varying network bandwidths

Challenges for HPC Software

- **Machine Independent Parallel Programming**
  - need for protection of programming investment
    - write once and tune, single source image
  - need for high performance on each target machine
    - close to hand coded parallel program in native programming interface
    - independent of algorithm choice
- **Ease of Use**
  - high level of abstraction
    - freeing programmer from gory details of managing complex hardware
    - increasing accessibility of parallelism
- **Programming Tools**
  - mechanisms for assisting in building, debugging and tuning parallel programs
    - user control from a high-level interface
- **Market Penetration**
  - must have a familiar environment on each commercial platform
  - users must accept languages and tools

HPC Software Successes

- **Compiler Memory Hierarchy Management**
  - register allocation, register blocking, cache blocking, cache prefetching
  - Memory reorganization for parallelism
    - reduction of false sharing
- **Compiler Extraction of Parallelism**
  - Automatic parallelization
    - effective for loops on shared-memory multiprocessors
  - Language and compiler support for data parallelism
    - HPF available on every parallel platform
- **Support for Portable Parallel Programming**
  - HPF, MPI, HPC++, OpenMP, Java
- **Parallel Libraries**
  - Communication, Math, Data Structures
- **Integrated Tools**
  - Performance Analysis and Tuning
  - Debugging

An Assessment of HPC Software Efforts

- **Research**
  - Substantive progress in software: languages, compilers, tools
- **Impact on End Users and Products**
  - Overall the harvest of compiler and tool technologies for scalable parallel computing has been meager, with few unqualified successes
    - Message-passing libraries (MPI, PVM)
    - Performance analysis and tuning tools
  - Progress in research has not been effectively transferred to practice
- **Implications**
  - Little support beyond MPI for machine-independence
  - Limited penetration of scalable parallelism in real applications
  - Divergence of programming models
    - Increased reliance on multiple-language solutions

CRPC

Reasons for Limited Success

- **Pace of architectural change**
  - shared memory, distributed memory, distributed shared memory, clusters, heterogeneous distributed networks
  - difficult to establish a general strategy
- **HPCC investment strategy counterproductive**
  - direct investment in software technologies inadequate
  - grand challenge applications focused on performance and results at the expense of technology development
- **Technology transfer mechanisms are flawed**
  - users want technologies that are mature, reliable, and standard
  - software technologies are complex
  - small size of high-end HPCC market
    - limited corporate resources for software investment
    - companies reluctant to gamble on new technologies
  - early deployments of new technologies are notoriously unreliable
    - users become disaffected

Future Architectures I

- **Petaflops/Petaops Architectures** **(year: 2007)**
  - Custom high-performance processors
    - superconductivity, multithreading
    - 10,000 at 100 gigaflops
  - Commercial off-the-shelf (COTS)
    - 100,000 at 10 gigaflops
  - Processor-in-memory
    - processor and memory on one chip
    - 1,000,000 at 1 gigaflops
- **Implications**
  - Deep memory hierarchy
    - up to 10 levels (thousands or tens of thousands of processor cycles)
  - High levels of parallelism
    - at least 10 million way parallelism must be found in the application used to exploit parallel processors and hide memory latency

Future Architectures II

- **Distributed Heterogeneous Systems**
  - geographically distributed high-end systems
  - piles of PCs
  - message passing across nodes, shared within
  - nodes and links have varying power
    - number and power of processors, bandwidth of links
    - network bandwidth varies with load
- **Implications**
  - Program decomposition
    - load balancing
    - matching function to architecture
    - minimizing distant communication
  - Interaction with system and network
    - system allocation
    - fault tolerance and migration
    - quality of service

Future Applications

- **Application Complexity**
  - Irregular, adaptive computation
  - Multidisciplinary simulation and design
  - Commercial applications
    - Java
  - Data intensive computation
- **Application Composability**
  - Applications will involve many programs
  - MADIC study
    - 10,000 applications
    - untrusting developers
  - Language interoperability
  - Application development tools

What We Must Do

- **Attack Performance Bottlenecks**
  - ameliorate the memory hierarchy problem
    - including I/O
  - find more parallelism
- **Support Portable High-Performance Computing**
  - develop and support standards
- **Raise the Level of Programming Abstraction**
  - abstract specification of parallelism
  - problem-solving environments, scripting languages
  - support for Interoperability
    - multiple languages, multiple applications
- **Rethink Compiler Design**
  - optimizations postponed until load and run time
  - integration of data and run-time information into compilation
  - integration of tool support
  - increased reliability through component structure
  - mechanisms to protect source code

Memory Hierarchy Management: Key Ideas

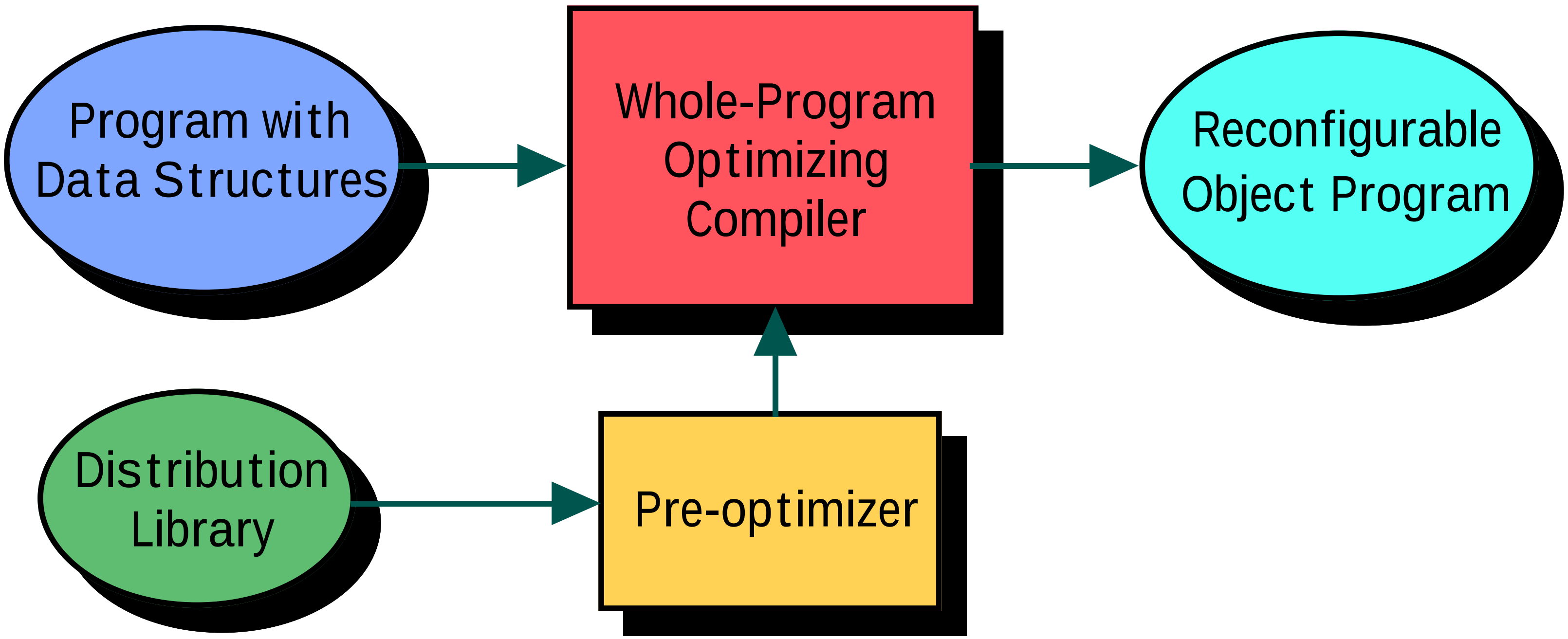
- **Program Reorganization**
  - register and cache blocking
  - loop splitting
- **Software Prefetching**
  - prefetch selection and placement
- **Memory Reorganization**
  - variable grouping on cache lines
  - array storage reorganization
  - dynamic reorganization schemes
- **Inclusion of I/O in Memory Hierarchy**
  - extension of cache techniques
    reorganization, prefetching
  - improvement factors in the hundreds

Distributed Heterogeneous Computing

- **Program Decomposition**
  - Distributed objects
  - Distributed data structures
  - Adaptive distribution of standard data structures
- **Scheduling**
  - Static and dynamic performance estimation
  - System performance parameterization
  - Adaptive load matching
- **Latency Management**
  - Interaction with Quality-of-Service facilities
  - Fast translation of data formats

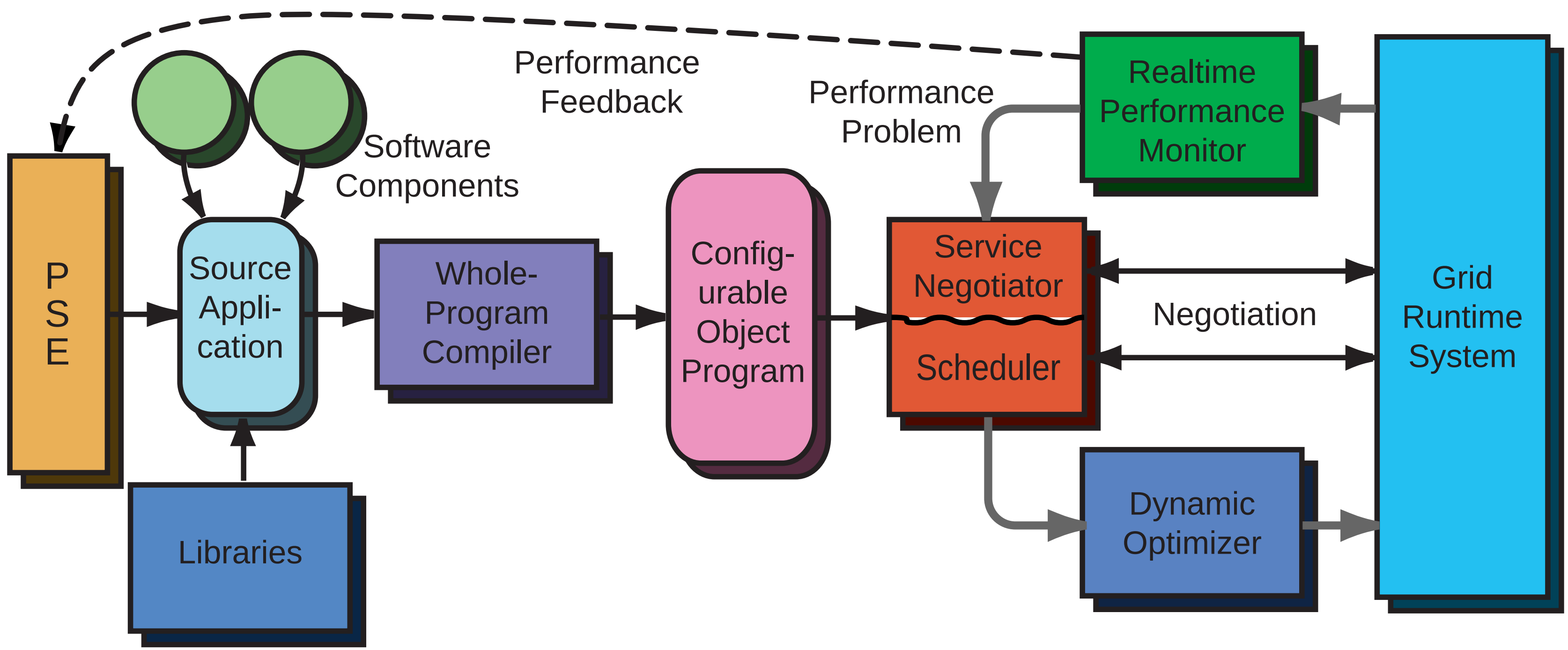
Reusable Distributed Programs

- **Goal: Separation of computation from distribution strategy**
  - user defines distribution explicitly in form of a library
  - compiler optimizes program with distribution for target architecture



Compilation for Heterogeneous Grids

- **Challenge**
  - dynamic, changing nature of target, difficult to manage by hand
- **Solution**
  - A new program preparation architecture



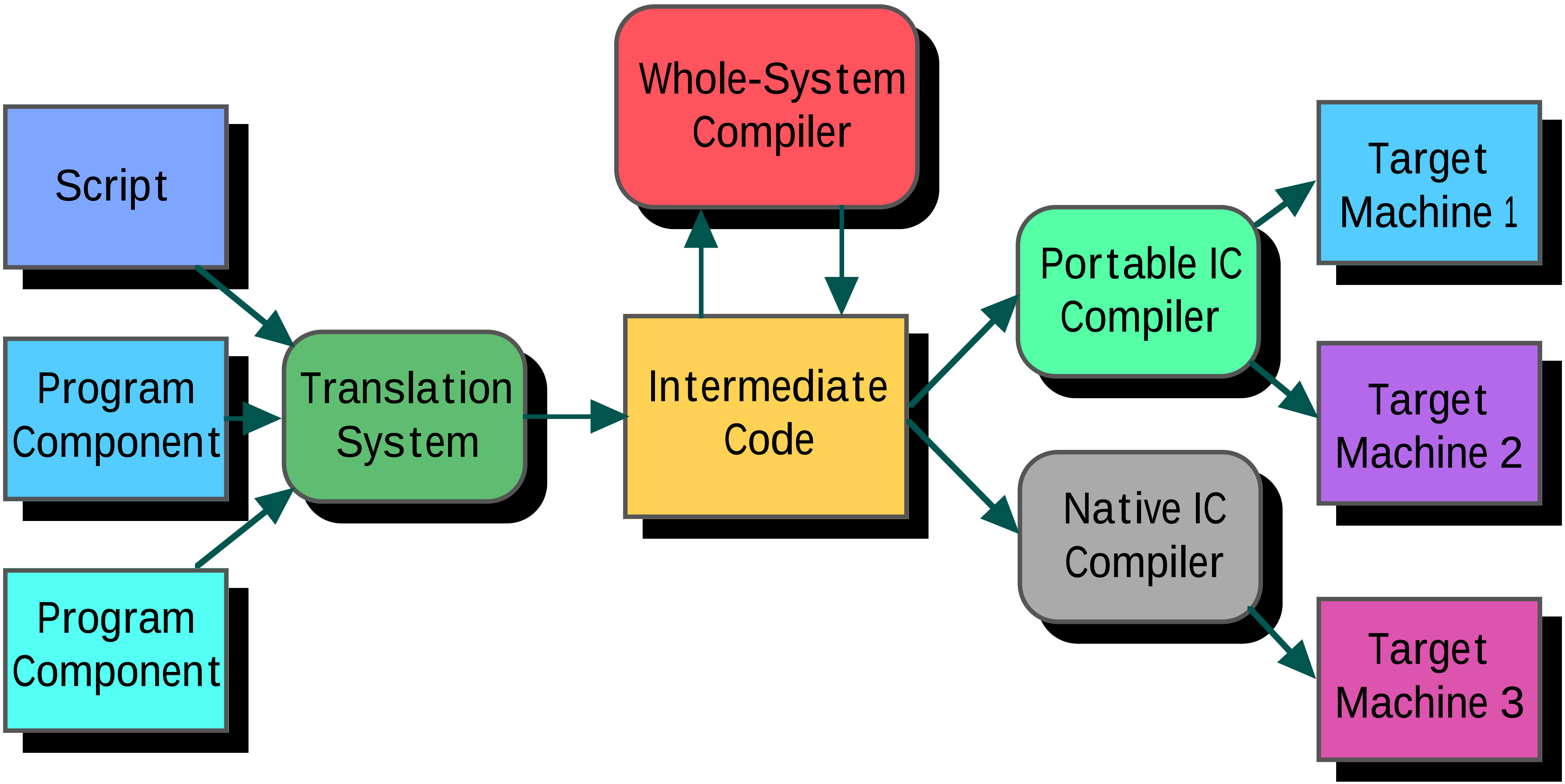
Programming by the End User

- **Challenges**
  - programming is hard
  - professional programmers are in short supply
  - high performance will continue to be important
- **A Solution: Make the End User a Programmer**
  - professional programmers develop components
    - languages: Java, Fortran, C++
  - users integrate components using:
    - problem-solving environments (PSEs)
    - scripting languages (possibly graphical)
      - examples: Tcl, Visual Basic, AVS, Khoros
- **Compilation for High Performance**
  - translate scripts and components to common intermediate language
  - optimize the resulting program using interprocedural methods

Script-Based Programming System

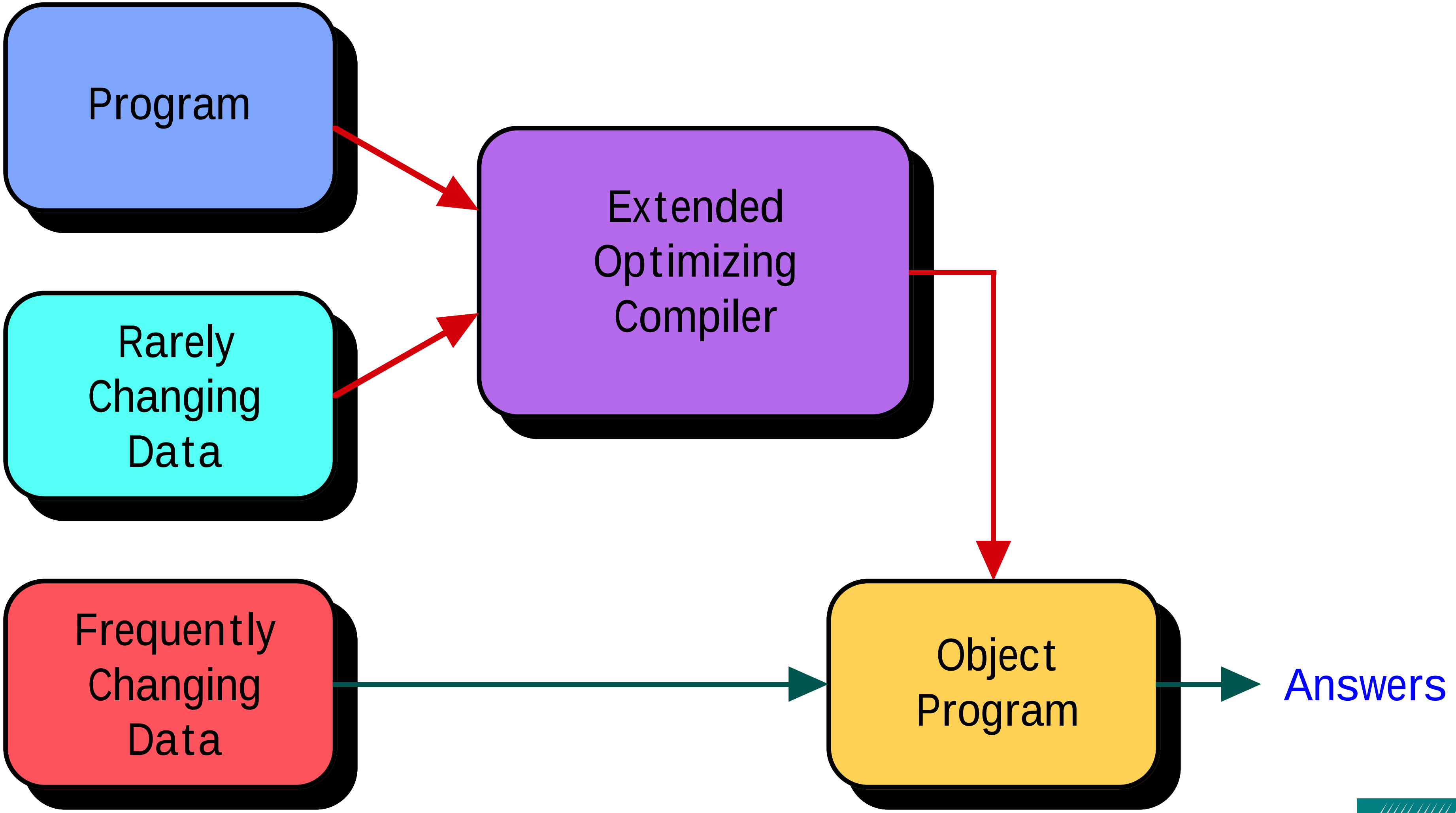
- **Compilation of all components to single intermediate code**
  - script, whole applications, all single-language components
  - intermediate language must support parallelism and memory hierarchy
- **Whole-system optimization of intermediate code**
  - interprocedural analysis
  - systematic inlining
- **Translation to high-performance parallel programs**
  - standard intermediate code + communication
- **Compilation to native machines**
  - portable or native compilers

Script-Based Programming System



CRPC

Compilation with Data



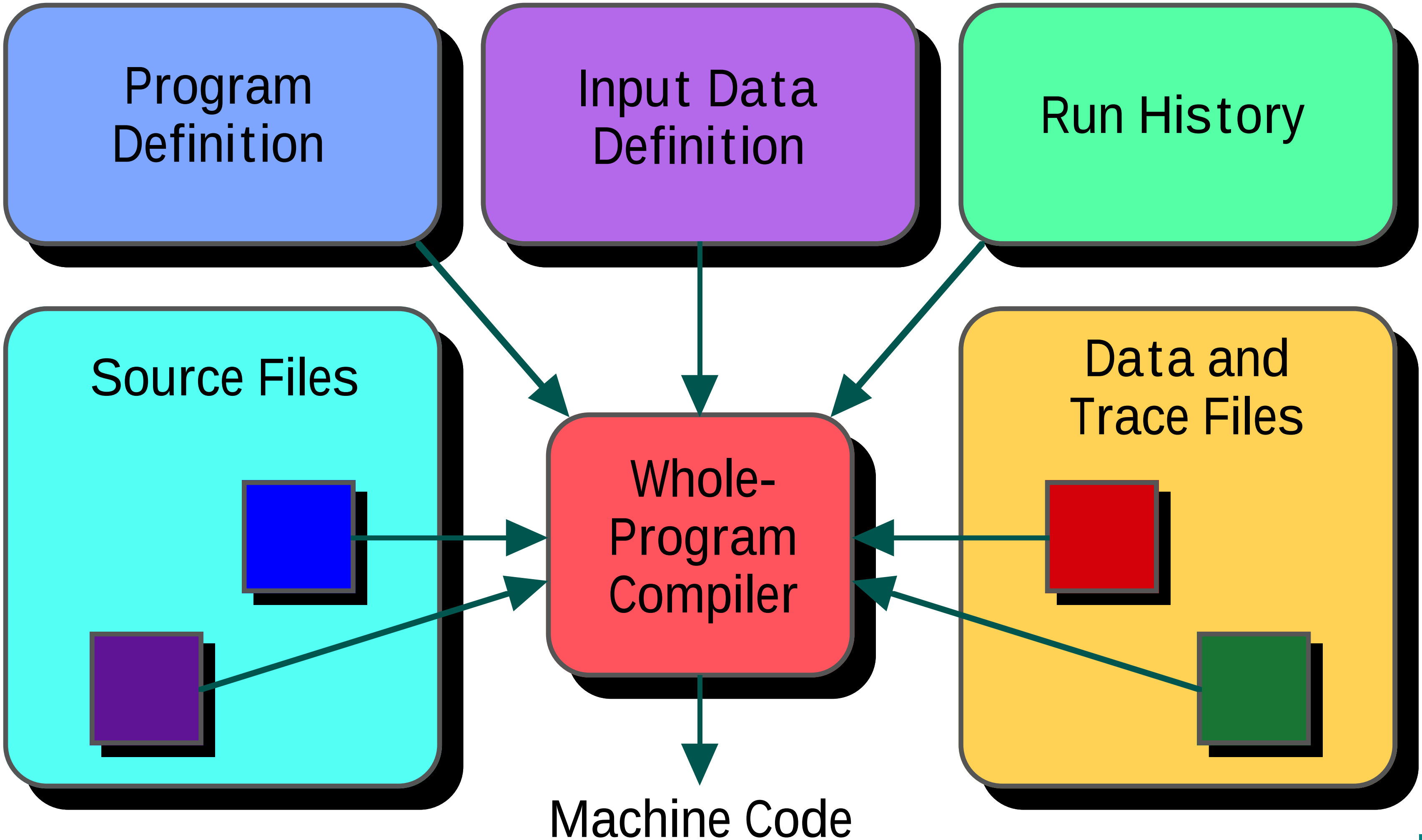
CRPC

New Compiler Architecture

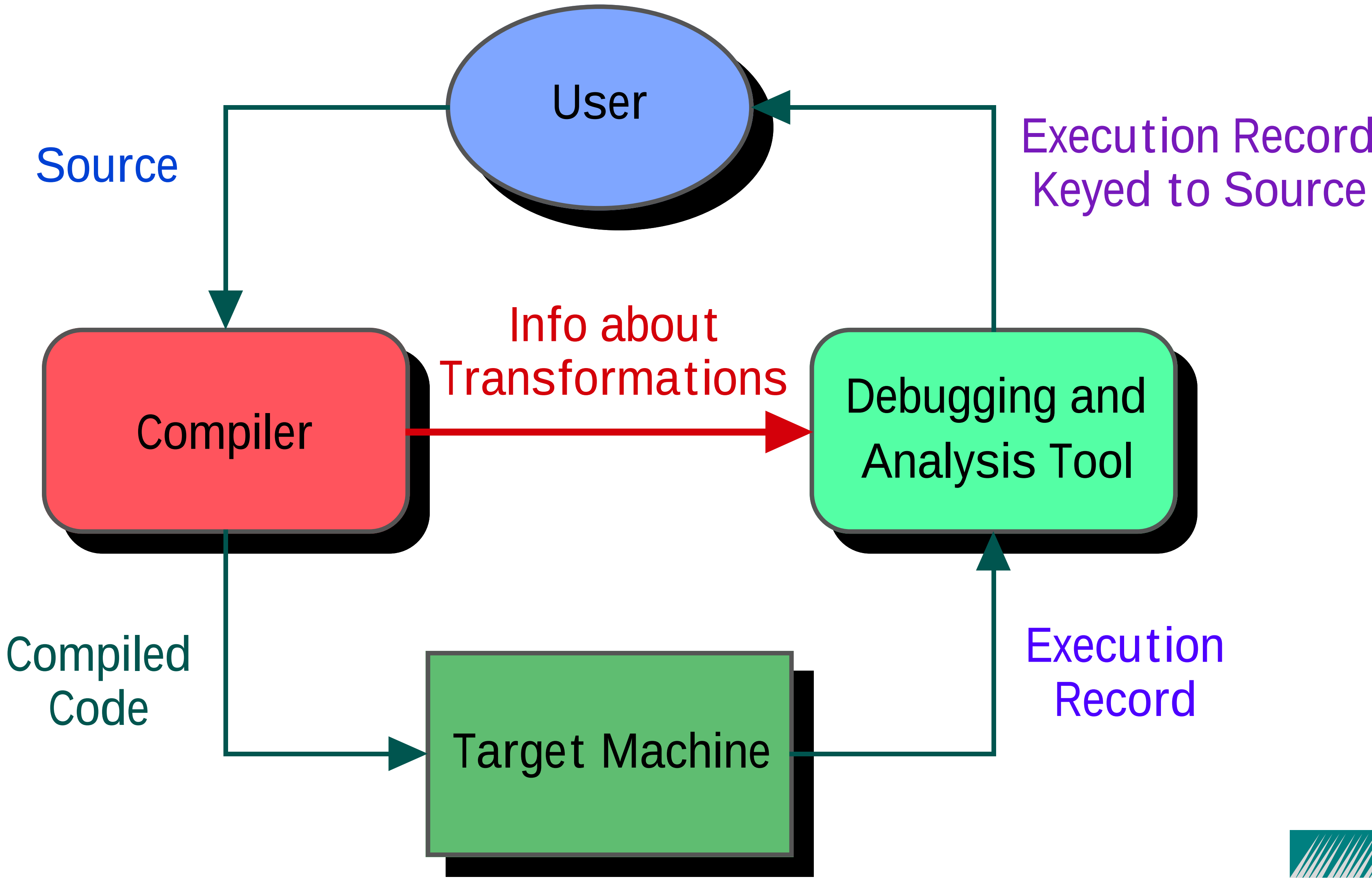
- **Flexible Definition of Computation**
  - Parameters
    - program scheme
    - subprogram source files (s_1, s_2, ..., s_n)
    - run history (r_1, r_2, ..., r_k)
    - data sets (d_1, d_2, ..., d_m)
    - target configuration
- **Compilation = Partial Evaluation**
  - may be several compilation steps
    - information available at different times
- **Program Management**
  - Must decide when to back out of previous compilation decisions in response to change
  - Must decide when to invalidate certain inputs
    - previous run histories

CRPC

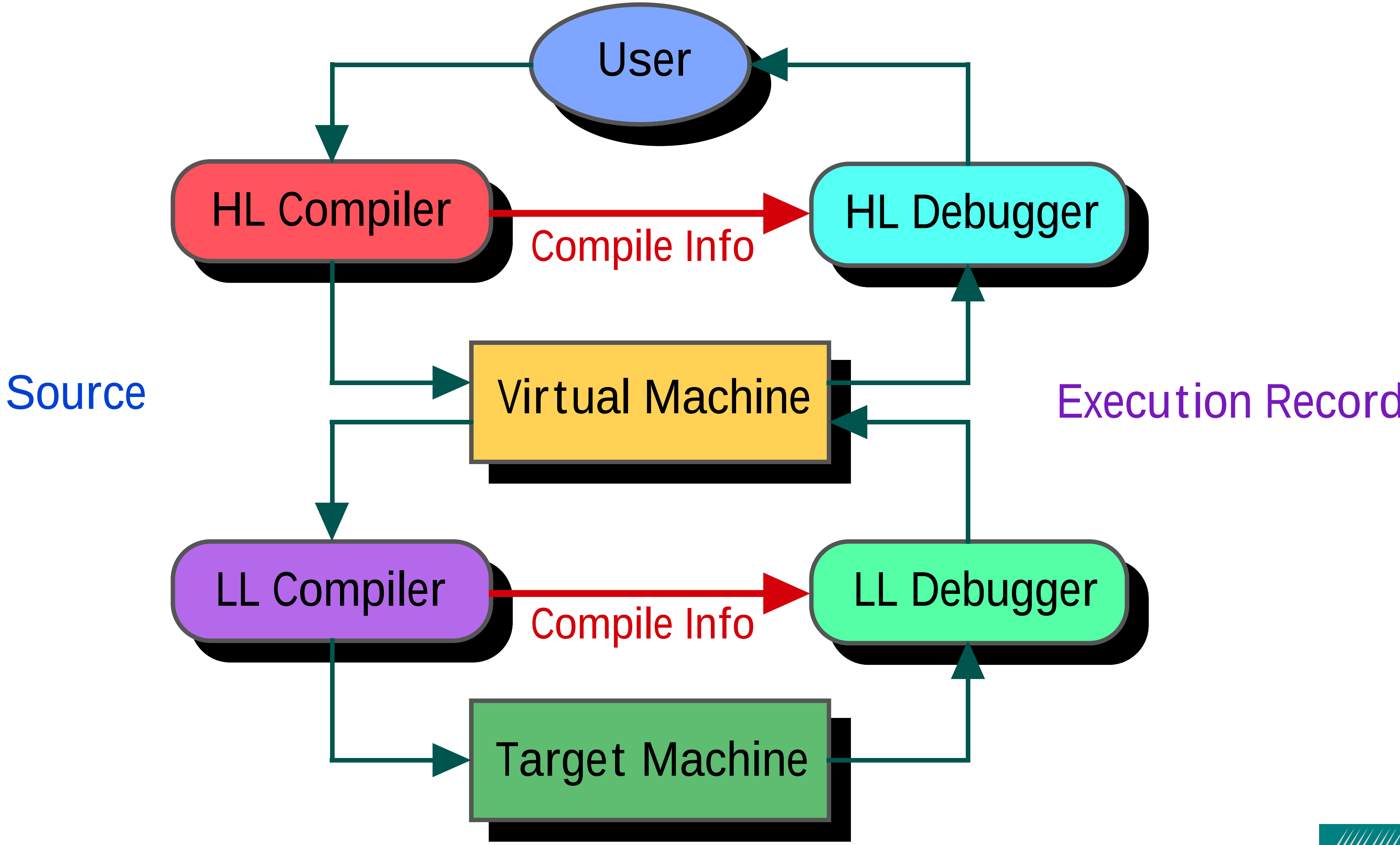
New Technology Program Environments



The Role of Tools



Composition of Tools



Challenges for Compilation Research

- **Complex Architectures**
  - more parallelism
  - deeper memory hierarchies
  - heterogeneous distributed systems
    - late binding of actual machine configuration
- **Complex Applications**
  - irregular, adaptive, dynamic computations
  - multiple programs, multiple languages, multiple parallelism styles
  - script-based system compositions
- **What We Must Do**
  - Attack performance bottlenecks
  - Support portable high-performance computing
  - Raise the level of programming abstraction
  - Rethink compiler design
    - support for performance, multilevel compilation, reliability, security